LUTRON

Introduction

This document is a quick start guide to Lutron Integration using LEAP (Lutron Extensible Application Protocol) and LAP (Local Access Pairing).

LAP is a service hosted on the Lutron processor at port 8083. It implements a protocol for the client device (e.g. your touchscreen) and the processor (the Lutron device) to exchange credentials, which will allow that client device to establish a secure and authenticated connection to that processor for LEAP API integration.

LEAP is a service hosted on the Lutron processor at port 8081. It implements an API for the client device to discover, monitor and control resources in the Lutron system (e.g. lighting zones, control keypads, etc.)

This document will walk you through how to:

1. Generate the keys needed to connect to the LAP service.
2. Find the Lutron processor on the network using mDNS.
3. Use the LAP service to exchange credentials.
4. Use the credentials from the LAP exchange to connect to the Lutron processor for LEAP API integration

You must have the following Lutron sample program files on your machine:

- `lap_sample.py` - walk through the process of establishing credentials with a specific Lutron system
- `leap_sample.py` - walk through accessing the Lutron API
- `discovery_sample.py` - show you how to find Lutron processors using Bonjour and mDNS
- `definitions.py` - common definitions
- `cert_checks.py` - useful diagnostics
- `app_sample.py` - shows you how to get basic information from your system

You must also have the following software available on your machine:

- OpenSSL
- dns-sd (available through the Apple Bonjour Dev Kit for Windows users)
- Python

A. Generate the keys needed to connect to the LAP service

Step 1: Create a developer private key and a certificate signing request

Generate the private key.

```
> openssl genrsa -out lap_private_key.pem 4096
```

Get `lutronelectronics.conf` from the Lutron developer portal. Update the parameter after `-subj` with information on your company and product. Create a certificate signing request (CSR).

```
> openssl req -new -key lap_private_key.pem -out lap_signing_request.csr -config
lutronelectronics.conf -subj "/C=US/ST=California/L=San
Francisco/O=FooCorp/OU=CoolProductPlus/CN=FooCorp Development"
```

Step 2: From the developer portal, get the certificate signed by Lutron and related certificates

Sign in to the Lutron developer portal, upload the CSR and request it be signed by Lutron. This will take about one business day. You will receive an update via email when it is ready for download.

From the developer portal download the following:

- The signed certificate from Certificate Requests - save it as `lap_signed_csr.pem`
- The Lutron root CA from Documents - save it as `lap_lutron_root.crt`
- The Lutron intermediate CA from Documents - save it as `lap_lutron_intermediate.pem`

B. Find the Lutron processor on the network using mDNS

Use Bonjour and mDNS discovery to find the link local address or IP address of a processor in the Lutron system you intend to integrate with.

Run `discovery_sample.py` to find the Lutron processors on your network.

```
> python .\discovery_sample.py
Searching for Lutron processors (5s)
  Server: Lutron-0577515a.local., IPv4: 169.254.125.237, System: myRoomProcessor,
Serial: 0577515A, MAC: a4:06:e9:6e:43:b4, Claimed: Unclaimed, Sw version:
23.04.25f000
  Server: Lutron-04e57558.local., IPv4: 10.5.12.150, System: HWQSProcessor,
Serial: 04E57558, MAC: 3c:e4:b0:1d:76:3d, Claimed: Claimed, Sw version:
23.02.22f000
  Server: Lutron-00b01654.local., IPv4: 10.5.12.146, System: AthenaProcessor,
Serial: 00B01654, MAC: 88:3f:4a:91:97:64, Claimed: Claimed, Sw version:
23.03.10f000
```

Use either the `Server` or `IPv4` as the address for the processor. For the rest of this guide, we will assume a static IP address for the processor. You can only connect to a processor that has been started up. If the `Claimed` state is `Unclaimed`, use the Lutron software to startup the processor before you try to connect.

C. Use the LAP service to exchange credentials

In this section, we will use the certificates and keys from the previous section to demonstrate how a client device (e.g. a touchscreen) will connect with the LAP service on the target Lutron processor to generate the keys needed to connect to the LEAP service. This is a one-time process for any client device - processor combination. On a production client device (e.g. a touchscreen), these will be saved locally on the client device and used to connect to the processor or reconnect to the processor after powerup.

Step 1 (optional): Test the certificates using openssl

Use the following command to test the certificates using OpenSSL. Use the IP address of your processor in the `-connect` parameter

```
> openssl s_client -connect 192.168.0.100:8083 -CAfile lap_lutron_root.crt -cert
lap_signed_csr.pem -key lap_private_key.pem -cert_chain
lap_lutron_intermediate.pem

CONNECTED(00000144)
depth=4 C = US, ST = PA, L = CB, O = Lutron Electronics Inc, OU = Lutron
Electronics Inc, CN = lutron-root
verify return:1
...
    Start Time: 1686320866
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: no
---
_
```

If the connection is not established or breaks immediately after connecting, check the following:

1. Are you using the correct pair of private key (`lap_private_key.pem`) and signed CSR (`lap_signed_csr.pem`)? Run `cert_checks.py` for basic validations.
2. Is your company authorized to connect to this Lutron system type?
3. Do you have the right IP address for a Lutron processor?
4. Have you setup and transfered a database to the Lutron processor?

In the OpenSSL session created above, type the following string to test the LAP connection.

```
{ "Header": { "RequestType": "Ping", "ClientTag": "xyzzy" } }
```

If you get the following response, you have successfully established a connection.

```
{"Header":{"StatusCode":"204 No Content","ClientTag":"xyzzy"}}
```

You can close this OpenSSL session now.

Step 2: Use the python script to walk through the LAP protocol

Run `lap_sample.py` in the directory with the files `lap_private_key.pem`, `lap_signed_csr.pem`, `lap_lutron_root.crt` and `lap_lutron_intermediate.pem`.

The script will show you how to:

1. Combine `lap_lutron_root.crt` and `lap_lutron_intermediate.pem` into one file `lap_lutron_chain`.
2. Create a device specific private key `leap_private_key` and a device specific signing request.
3. Establish a secure connection with the Lutron processor on port 8083
4. Have the user prove physical access to the processor by pressing a button on the processor
5. Retrieve the Lutron processor specific CA `leap_lutron_proc_root.pem`.
6. Get the processor to sign your CSR and retrieve the signed CSR `leap_signed_csr.pem` and the intermediate processor specific CA `leap_lutron_proc_intermediate.pem`.

The script will save all certificates and keys that will be needed for the LEAP API connection in the same folder with the `leap_` prefix.

D. Use the credentials from the LAP exchange to connect to the processor for LEAP API integration

In this section, we will use the certificates and keys from the LAP exchange, in the previous section, to connect to the LEAP API service (port 8081).

Step 1 (optional): Test the certificates using openssl

Use the following command to test the certificates using openssl. *Use the ip address of your processor in the -connect parameter*

```
> openssl s_client -connect 192.168.0.100:8081 -CAfile lap_lutron_root.crt -cert
leap_signed_csr.pem -key leap_private_key.pem

CONNECTED(00000144)
depth=4 C = US, ST = PA, L = CB, O = Lutron Electronics Inc, OU = Lutron
Electronics Inc, CN = lutron-root
verify return:1
```

```
...
    Start Time: 1686320866
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: no
---
_
```

If the connection is not established or breaks immediately after connecting, check the following:

1. Are you using the correct pair of private key (`leap_private_key.pem`) and signed CSR (`leap_signed_csr.pem`)? Use `cert_check.py` for basic certificate checks.
2. Do you have the right IP address for the Lutron processor?

In the openssl session created above, type the following string to test the LEAP connection.

```
{ "Header": { "RequestType": "Ping", "ClientTag": "xyzzy" } }
```

If you get the following response, you have successfully established a connection.

```
{"Header":{"StatusCode":"204 No Content","ClientTag":"xyzzy"}}
```

You can close this openssl session now.

Step 2: Use the python script to walk through the LEAP protocol

Before you run the following scripts, edit `definitions.py`. Set `LUTRON_PROC_SERIAL`, `LUTRON_PROC_MAC`, `LUTRON_PROC_ADDRESS` to values you get from the running discovery for the processor you want to connect to. Set `LUTRON_PROC_SYSTEM_TYPE` to the system type for the processor.

Run `leap_sample.py` in the directory with the files `leap_private_key.pem`, `leap_signed_csr.pem` and `lap_lutron_root.crt`.

The script will show you how to:

1. Establish a secure connection with the processor on port 8081
2. Read the root area of the system to prove you are authenticated

The script will show you the response to the API request to read the root area. If you see it, you have successfully connected to the Lutron processor. The files `leap_private_key.pem`, `leap_signed_csr.pem` and `lap_lutron_root.crt` may be used to open a connection at startup or reconnect to the Lutron processor as needed. You don't need to run the LAP protocol again unless you are replacing the client device (e.g. a touchscreen) or the Lutron processor.

tt tn xx c x x a F i x fk 9 c fkffi fk fk
cfk x fk c F fk fk tt fk fk F a tn fk i x a x
ffi c i fk fk tt fk F fk ffi fk
